Miniproject Report

TTK4147 - Real-time Systems

Group 14:
Øystein Brox
Morten Jansrud

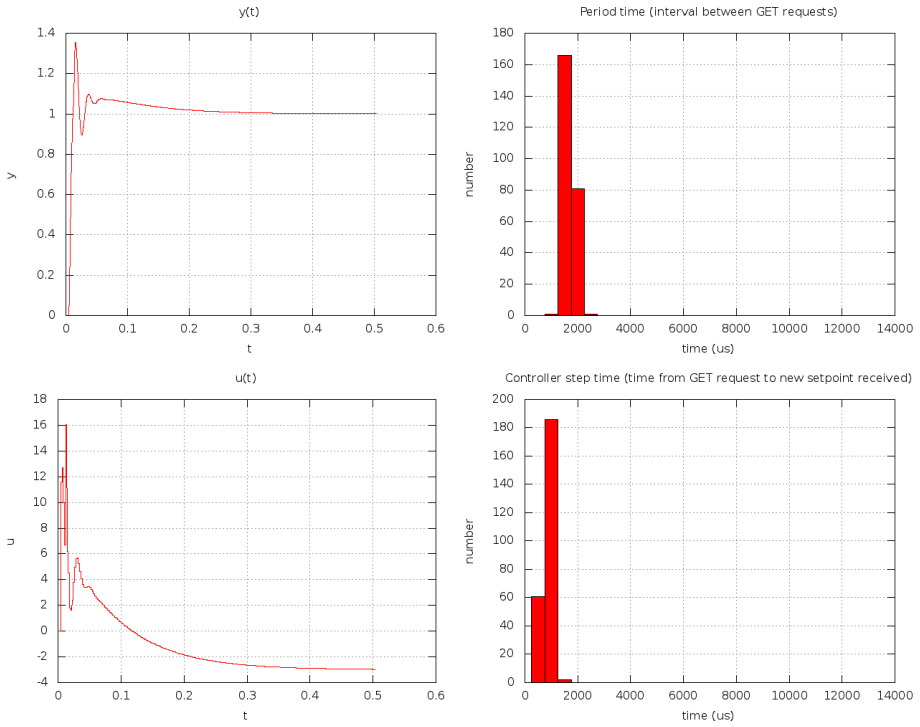
Department of Engineering Cybernetics
NTNU
Norway
November 2017

Task 1

Design

The project structure consists of three separate C-files. One Client file that contains functionality used by both part 1 and 2, the functions of starting and stopping the server and a main function that starts either part based on input arguments. Additionally one separate file for each part of the assignment.

The period of the regulator and the desired program run time is defined in the header file. Based on these two variables we calculate the number of regulator iterations.



Results

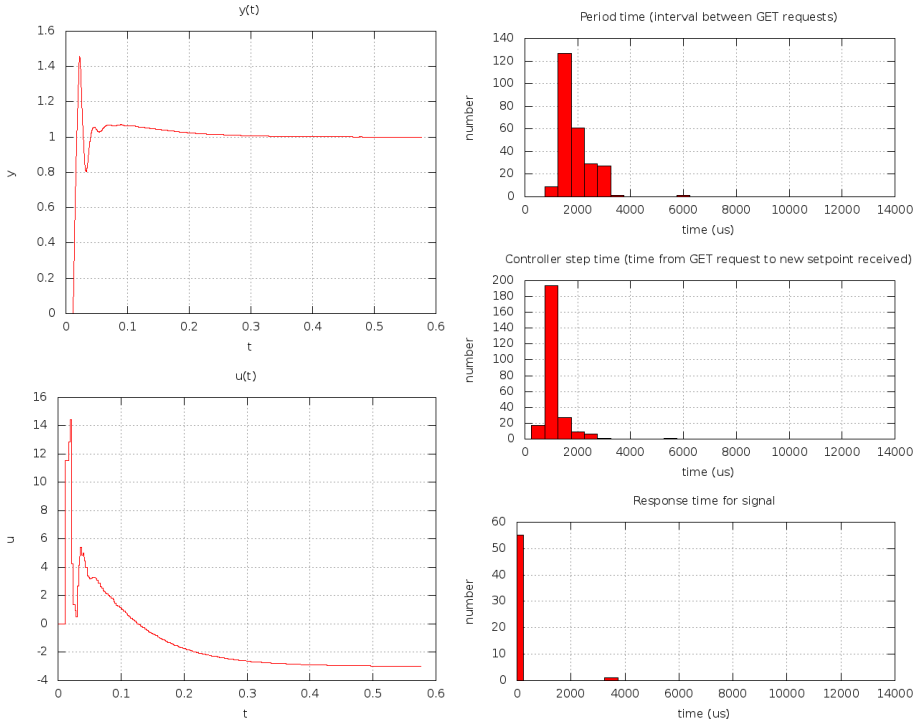
A period of 2ms gave the best results based on our tests. The step response stabilizes close to the reference after 0.2 seconds which we consider quite good. We haven't made a huge effort to optimize the controller .

Task 2

Design

We have a mutex in order to protect the UDP calls which prevents different threads from sending simultaneously. The mutex is an additional redundant protection as UDP is thread safe in POSIX, however, we chose to include the feature as it was specified in the assignment. This addition didn't influence the performance or add any noticeable overhead.

Semaphores was created in order to notify of new y-values and signals that needed acknowledgement. We had a mutex to protect the read/write of the y-value to make absolute certain that this was an atomic action.



Results

The step response is almost identical to task 1 and we can conclude that the signal acknowledgement didn't noticibly interfere with the regulator performance.

Almost every signal gets an acknowledge immediately, however there are a few with a little longer response time. With a more optimized code they might have been acknowledged faster. Without priority of the threads it's impossible to guarantee that every signal gets an immediate response.